

**NORTi TCP/IP 用
Intersil Prism2.x
無線 LAN ドライバ
取扱説明書**

第一版 2003 年 2 月 27 日

レッドウェーブテクノロジー株式会社

目 次

1. はじめに.....	3
2. ファイル構成.....	3
3. 移植の際に修正が必要なファイル	5
4. コンパイル、リンク	10
5. 設定変更用 API.....	10
6. シェルコマンド(無線 LAN 追加分)の説明	16
7. 制限.....	19

1. はじめに

本ドライバは、Intersils 社の Prism2.x 無線 LAN カードを使用して、NORTi Version4 のプロトコルスタックを利用できるようにするものです。

2. ファイル構成

本ドライバは、NORTi での DP83902(NS)や MB86964(富士通)などのドライバと同様に特定のボード用のソースの一部として提供いたします。

今回提供する「特定ボード」は株式会社エンベデッドシステム製 EyeNodeSH3 です。

本ドライバが、NORTi のサンプルとして無償提供されるわけではありません。

ディレクトリツリーでいうところの位置づけ上、そうしているという意味です。

なお、ドライバ周辺は PCMCIA 経由での CHIP 制御となりますので、カード制御部さえ書き換えれば、ほぼそのままどのような環境においても動作可能になります。

以下に、ファイル構成を示します。

今回の提供環境は、C:\NORTi\SMP\SH\EyeNode_with_Prism2 におかれることを想定しています。

<ドライバ関係ヘッダファイル>

hfa3841.h	prism2 chip の基本的な定義、hfa3841.c のプロトタイプ等
prism2.h	prism2.c のプロトタイプ宣言
wlan_api.h	wlan_api.c のプロトタイプ宣言
wlan_com.h	無線 LAN 特有の定義等
wmrshpc.h	PC カード制御用の基本的な定義、wmrshpc.c のプロトタイプ等
prism_util.h	prism_util.c のプロトタイプ宣言

<ドライバ関連ソースファイル>

hfa3841.c	hfa384x(PRISM2.x の型番)のコアな制御関数群
nonet_prism2.c	NORTi Network 使用のドライバインターフェイス関数群
prism2.c	hfa384x(PRISM2.x の型番)のコアな制御関数群(その2)
prism_util.c	ディレイ等のユーティリティ
wlan_api.c	PRISM の設定を変更するための API 関数群
wmrshpc.c	PCMCIA カード制御関数群

<サンプル環境関連ソースファイル>

nos16550.h MS7709ASE01 用サンプルと同じです。

nos7709a.h 同上

ms7709.h 同上

nonecfg.c 同上

initsh.c 同上

n4i7709a.c 同上

nos16550.c 同上

nos7709a.c 同上

sh7709a.h 同上

net7709a.c MS7709ASE01 用サンプルに以下のものを追加しました。
無線 LAN 関係の変数、定数(設定関連)の追加
Chip 設定をいじるコマンドを SHELL コマンドに追加
MainTask に PC カード初期化と無線 LAN カードの初期化を追加
これらは tcp_ini の前に行う必要があります。
ini_mpu()の内容を追加：EyeNodeSH3 はカードを使うのに面倒な設定が必要になります。

vec7709ah.asm CPU 内蔵周辺回路の初期化定数を EyeNode 用に変更しました。

<デバッガ用>

little.cpf Computex H-UDI SH7709A/SH7729 用プロジェクトファイル

Init_EyeNode_SH3.mcr 初期設定用マクロファイル
H-UDI や PALMiCE の初期化マクロとして使用してください。

<ディレクトリ Shc5> 日立 C 5.1C 用コンパイルディレクトリ

net7709alh.mak MAKE ファイル

net7709alh.sub リンク用サブファイル

・EyeNodeSH3 はリトルエンディアン専用なので、BigEndian 用 Make ファイルは作成していません。(ドライバ自体の BigEndian での動作は別環境で確認済みです)。また SHC6 用、SHC7 用コンパイル環境は作成中です。

3. 移植の際に修正が必要なファイル

(1) wmrshpc.h

PC カードコントローラが違うものになった場合、それ用に変更が必要です。

<ウィンドウアドレス>

PC カードコントローラ自体が配置されているアドレスや、PC カードコントローラに対する初期設定により、アドレスが変更になりますので、正しく修正してください。

```
#define MEM_WIN      0xb7000000      /* Real Word Memory Window Area    */
#define ATR_WIN      0xb6000000      /* Real Word Attribute Window Area  */
#define IO_WIN       0xbbf00000      /* Real Word I/O Window Area        */
```

<レジスタ定義等>

こちらでは関知しません。

wmrshpc.c に用意する関数の中で必要な定義を行ってください。

<プロトタイプ>

wmrshpc.c には、このプロトタイプにあった関数を作成してください。

(2) wmrshpc.c PCMCIA カード制御関数群

• UH wPCCard_check(void)

カードの存在 Check を行います。

• ER wPCCard_start(void)

カードが挿入されていれば、電源を ON します。

ハードによりますが、カードがサポートする電源電圧をチェックし、その電圧を越えた電源電圧を与えないで下さい。

そのあと、リセット出力を所定の時間行ってください。

• static ER wPCCard_end(void)

カード電源を OFF します。

本ドライバでは OFF することはありませんので使用しません。

• ER wPCCard_init(void)

ユーザが初期化時に呼び出す関数です。

この中で、wPCCard_start()と wInit_MR_SHPC()を呼び出して下さい。

その後割り込みハンドラを定義してください。

割り込みコントローラ等を操作して割り込みを許可する必要がある場合は、ハンドラの定義後に行ってください。

• void wPCCard_CopyCIS(UB *buf, int cnt)

タプルの情報を buf にコピーします。

この関数に変更の必要がありません。

- **void wInit_MR_SHPC(void)**

なにか追加で初期化を行いたい場合に使用します。

EyeNode 用では LED を点灯しています。

- **ER wPCCard_HDRregist(UW cardhdr)**

この関数に変更の必要がありません。

- **static void CARDinterrupt(void)**

割り込みハンドラの入り口です。

NORTi の acre_isr を使用していますので void 型になっています

ここから Cardhdr ポインタを参照して、最終的に lan_intr() に飛ぶように作成してください。

(3) net7709a.c

/*--- WLAN ---*/で囲まれた所を参照してください。

・無線 LAN 関係の変数、定数(設定関連)の追加

MainTask()において、PC カードの初期化が終わったあと、

```
prdesc = prism2_init(IO_WIN,&config_data,NULL);
```

として、デバイスドライバの初期化を行っています。

関数 prism2_init()の仕様は以下のとおりです。

```
prism2_t * prism2_init(UW iobase,PRISMCFG *cfg,PRISM_FP func) */
```

iobase : ベースアドレス

cfg : コンフィグレーションテーブル

func : テーブルで設定できる以外の設定を行う関数へのポインタ

戻り値:

NULL : 初期化失敗

それ以外 : ディスクリプタアドレス

ここでいうコンフィグレーションテーブルを定義してください。この値を初期値としてコンフィグレーションされます。

また func には、API(後述)で用意した以外の処理を追加的に行いたい場合に指定して

下さい。NULL なら追加処理はないものとします。

本サンプルではコンフィグレーションテーブルは下記の様になっています。

```
/* 設定値 */
PRISMCFG config_data = {
    /* BASIC */
    1, /* UH NetworkType -> 0:Ad-Hoc/1:Infrastructure/3:Ad-Hoc Demo */
    1, /* UH Channel -> 1~14 までのチャネル */
    {"01234567890123456789012345678901"},
    /* UB SSID[32] -> 32 文字までの(0-9,A-F) ASCII データ */
    /* Rate setting */
    0x0003, /* UH BasicRate -> Bit0:1Mbps/Bit1:2Mbps/Bit2:5.5Mbps/Bit3:11Mbps */
    0x0008, /* UH SupportedRate-> Bit0:1Mbps/Bit1:2Mbps/Bit2:5.5Mbps/Bit3:11Mbps */
    0x0008, /* UH DataRate -> Bit0:1Mbps/Bit1:2Mbps/Bit2:5.5Mbps/Bit3:11Mbps */
    /* WEP setting */
    0, /* UH WEPType */
    0, /* UH WEPDefaultKeyID -> 0~3 までの WEP キー番号 */
    { 0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33 }, /* UB WEPKey0[13] */
    { 0x32,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33 }, /* UB WEPKey1[13] */
    { 0x33,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33 }, /* UB WEPKey2[13] */
    { 0x34,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33 }, /* UB WEPKey3[13] */
    3, /* UH AuthenticationType ->1:Open system/2:Shared Key/ 3:1+2 */
    2304, /* UH MaxDataLength -> 350~2304 までのデータ長 */
    2346, /* UH FragmentThreshold -> 256~2346 までの偶数値 */
    2432 /* UH RTSThreshold -> 0~3000 までの偶数値 */
};
```

• Chip 設定をいじるコマンド

設定変更用 API の使用サンプルとして追加してあります。

必要に応じて追加、変更、削除してください。

• MainTask()の先頭部で PC カード初期化と無線 LAN カードの初期化

PC カード制御関数の移植がうまく行えていれば、変更の必要は無いはずです。

• ini_mpu()の内容を追加

EyeNodeSH3 ではカードを使うのに RTC を初期化するなど、面倒な処理が必要です。

vec7709ah.asm に入れなくても良いが、なるべく早いうちに初期化すべき内容はここに記述してください。

4. コンパイル、リンク

DOS プロンプト(コマンドプロンプト)での MAKE(NMAKE.EXE)で行ってください。

SHC6,SHC7 用の MAKE に対応中ですので、いずれ作成しますが、MAKEFILE 中のコンパイルスイッチ等を参考に設定していただければ、Hew 等の WorkSpace でもコンパイルは可能です。

(1)MAKEFILE(net7709alh.mak)の説明

・TICK マクロを追加しました。これを変更すると各コンパイルに-def=MSEC=\$(TICK)を与えていますので所望の Tick での動作を行うようにコンパイルが可能です。

なお、NORTi の TCP/IP ライブラリは Tick は標準の 10mSEC でコンパイルされていますので、LIB 生成用の MAKE ファイルを修正して再生成が必要です。

・ENDIAN = LITTLE_ENDIAN

EyeNodeSH3 はリトルエンディアン専用なので、リトルエンディアン用の MAKE しか添付していませんが、ENDIAN=BIG_ENDIAN とすることによって、すべてビッグエンディアンにすることが可能です。

この場合、LINK 用のサブファイルで指定してるライブラリ(標準、カーネル、TCP/IP)はビッグエンディアン用のものに変更してください。

5. 設定変更用 API

• wlanapi_TxRates

機能：デバイスディスクリプタに Config 値を設定する

Config 値=0xFC84 にセットする送信 Rate

プロトタイプ:

```
int wlanapi_TxRates(hw,UH rate);
```

prism2_t *hw デバイスディスクリプタ

UH rate 設定レート

戻り値

エラーコード：E_PAR パラメータ NG

E_OK 正常終了

• wlanapi_SuppotedRates

機能：デバイスディスクリプタに Config 値を設定する
Config 値=0xFCB4 にセットする Supported Rates

プロトタイプ:

```
int wlanapi_SuppotedRates(hw,UH rate);  
    prism2_t *hw    デバイスディスクリプタ  
    UH rate         設定レート
```

戻り値

エラーコード：E_PAR パラメータ NG
 E_OK 正常終了

• wlanapi_BasicRates

機能：デバイスディスクリプタに Config 値を設定する
Config 値=0xFCB3 にセットする BroadCast/MultiCast/Mgmt の送信 Rate

プロトタイプ:

```
int wlanapi_BasicRates(hw,rate);  
    prism2_t *hw    デバイスディスクリプタ  
    UH rate         設定レート
```

戻り値

エラーコード：E_PAR パラメータ NG
 E_OK 正常終了

• wlanapi_Wep

機能：デバイスディスクリプタに記憶された Config 値を設定する

コンフィグ内容： RID= 0xFC28 WepFlag
 RID= 0xFC24 WepKey #0
 RID= 0xFC25 WepKey #1
 RID= 0xFC26 WepKey #2
 RID= 0xFC27 WepKey #3

プロトタイプ:

```
int wlanapi_Wep(hw,UH weptype,*Key0,UB *key1,UB *key2,UB *key3);  
    prism2_t *hw    デバイスディスクリプタ  
    UH weptype       WEP 選択 0:なし、1:64Bit、2:128Bit
```

※注 weptype 変数によって、実際に設定する WEPFlag とキー長を設定関数側で決めている。weptype という設定項目は PRISM2 自体には存在しない

UB *key0~*key3 WEP キー

戻り値

エラーコード：E_PAR パラメータ NG

E_OK 正常終了

• wlanapi_WepDefaultKeyID

機能：デバイスディスクリプタに Config 値を設定する
Config 値=0xFC23 にセットする、送信時使用 WepKeyID

プロトタイプ:

```
int wlanapi_WepDefaultKeyID(hw,UH keyid);
    prism2_t *hw      デバイスディスクリプタ
    UH keyid          キーID
```

戻り値

エラーコード：E_PAR パラメータ NG
E_OK 正常終了

• wlanapi_AuthType

機能：デバイスディスクリプタに Config 値を設定する
Config 値=0xFC2A にセットする認証タイプ

プロトタイプ:

```
int wlanapi_AuthType(hw,UH authtype);
    prism2_t *hw      デバイスディスクリプタ
    UH authtype      認証タイプ 1:Open System 2:Shared Key 3:1+2
```

戻り値

エラーコード：E_PAR パラメータ NG
E_OK 正常終了

• wlanapi_SSID

機能：デバイスディスクリプタに Config 値を設定する

コンフィグ内容：RID= 0xFC02 Desired SSID

RID= 0xFC04 OwnSSID

プロトタイプ:

```
int wlanapi_SSID(hw,UB *ssid);
    prism2_t *hw      デバイスディスクリプタ
    UB *ssid;          SSID(NULL 終端文字列)
    この API でセットした値は FC02,FC04 両方にセットされます
```

戻り値

エラーコード：E_PAR パラメータ NG
E_OK 正常終了

• wlanapi_PortType

機能：デバイスディスクリプタに Config 値を設定する

Config 値=0xFC00 の PortType を設定する

プロトタイプ:

```
int wlanapi_PortType(hw,UH porttype);
    prism2_t *hw    デバイスディスクリプタ
    UH porttype     0:Ad-Hoc/1:Infrastructure/3:Ad-Hoc Demo
```

戻り値

エラーコード：E_PAR パラメータ NG

E_OK 正常終了

• wlanapi_OwnChannel

機能：デバイスディスクリプタに Config 値を設定する

Config 値=0xFC03 の OwnChannel

プロトタイプ:

```
int wlanapi_OwnChannel(hw,UH channel);
    prism2_t *hw    デバイスディスクリプタ
    UH channel      使用 Ch
```

戻り値

エラーコード：E_PAR パラメータ NG

E_OK 正常終了

• wlanapi_FragThresh

機能：デバイスディスクリプタに Config 値を設定する

Config 値=0xFC82 フラグメント閾値をセットする

プロトタイプ:

```
int wlanapi_FragThresh(hw,UH thresh);
    prism2_t *hw    デバイスディスクリプタ
    UH thresh       閾値(256～2346 の偶数)
```

戻り値

エラーコード：E_PAR パラメータ NG

E_OK 正常終了

• wlanapi_RtsThresh

機能：デバイスディスクリプタに Config 値を設定する

Config 値=0xFC83 RTS/CTS 閾値をセットする

プロトタイプ:

```
int wlanapi_RtsThresh(hw,UH thresh);
    prism2_t *hw    デバイスディスクリプタ
    UH thresh       閾値 0～3000 までの偶数値
```

戻り値

エラーコード：E_PAR パラメータ NG
 E_OK 正常終了

• wlanapi_Getcfg_general

機能：config/info. 項目取得(汎用)

API にない項目取得を行う

プロトタイプ:

```
int wlanapi_Getcfg_general(prism2_t *hw, UH rid, void *buf, UH len)
    hw   : デバイスディスクリプタ
    rid  : RID 番号
    buf  : 書き込み先
    buflen : バッファ長(バイト数)
```

戻り値:

エラーコード：E_OBJ RID のデータ長が一致しない
 E_OK : 正常終了
 E_TMOUT: コマンド未完了(タイムアウト)

注意:

buf は、偶数番地から配置されていること

buflen は、偶数(バイト数)を指定すること/sizeof(buf)であること

buflen が、読み出したサイズより小さい場合はバッファに入れられるところまでいれて、E_NOMEM を返す。

バッファには、Length、RID も読み出したものすべてが格納される。つまり

```
buf[0]  Record Length( in Words,buf[1]~end of data )
buf[1]  RID
buf[2]  Data
buf[3]   :
buf[4]   :
buf[5]   :
```

★ワードイメージで記述しています。

なお、バッファに格納されたデータは、読んだ状態のまま格納しているので Stream でない Integer なデータはエンディアンにあわせてスワップが必要です。(リトルエンディアンの場合はスワップ不要)

• wlanapi_Getcfg_general

機能：config/info. 項目設定(汎用)

API にない項目設定を行う

プロトタイプ:

```
int wlanapi_Getcfg_general(prism2_t *hw, UH rid, void *buf,UH reclen)
```

hw : デバイスディスクリプタ

rid : RID 番号

buf : セットされるデータのポインタ

reclen : レコード長(ワード数)

戻り値:

エラーコード: E_OBJ RID のデータ長が一致しない

E_OK 正常終了

E_TMOUT コマンド未完了(タイムアウト)

注意:

reclen は、length フィールドに書き込むサイズそのものであるなので、RID とデータ本体のサイズを加算したものを渡してください。

buf[0] Record Length

buf[1] RID

この 2 つについては、本関数内で適切にバイトスワップを行って設定する。

ユーザは buf[2]以降を適切にセットすること。buf[2]以降は本関数内ではスワップなどは行わない。

※データの Format により、スワップが必要な場合とそうでない場合があるため。

String 等はスワップしては駄目

• wlanapi_Setcfg_Defaults

機能：一括設定

プロトタイプ:

```
int wlanapi_Setcfg_Defaults(prism2_t *hw)
```

hw : デバイスディスクリプタ

戻り値:

エラーコード: E_OK 正常終了

E_TMOUT コマンド未完了(タイムアウト)

注意:

既出の API を使用して、デバイスディスクリプタ上にセットしたコンフィグレーションをまとめて再設定する。

このとき、Static/Dynamic Configuration Entity の区別なく、一旦 Prism2 を Disable にし、一括設定を行い、Enable する。

なお、このとき呼び出しているのは wlan_Setcfg_Defaults なので、ユーザによる追加初期化関数を呼び出すことに注意

また、汎用設定関数で前もって設定した内容も、Static Configuration であれば、一度 Disable して Enable になることにより、Firmware にコピーされるので、反映される。

6. シェルコマンド(無線 LAN 追加分)の説明

設定変更用 API の使用サンプルとして作成したが、動作確認用としても使用できるので、簡単に説明する。

(1)ch コマンド：無線 LAN 使用 Ch を設定します。

>ch channel_number

channel_number には 1～14 の数値を指定してください。

設定は recfg コマンド実行後に反映されます。

(2)port コマンド：ポートタイプを設定します。

>port port_type

port_type には、0,1,3 のいずれかを指定してください。

0: 802.11 AdHoc

1: Infra Mode

3: 擬似 AdHoc

設定は recfg コマンド実行後に反映されます。

(3)ssid コマンド:ssid を設定します。

>ssid ssid_string

ssid_string には、32 文字までの SSID 文字列を指定してください。
設定は recfg コマンド実行後に反映されます。

(3)wep1 コマンド: 128Bit WEP のサンプル設定を行います。

>wep1

サンプルとして、以下の WEP キーを設定します。

設定は recfg コマンド実行後に反映されます。

key0: 0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xaa,0xbb,0xcc,0xdd

key1: 0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2a,0x2b,0x2c,0x2d

key2: 0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3a,0x3b,0x3c,0x3d

key3: 0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4a,0x4b,0x4c,0x4d

(4)wep2 コマンド: 64Bit WEP のサンプル設定を行います。

>wep2

サンプルとして、以下の WEP キーを設定します。

設定は recfg コマンド実行後に反映されます。

key0: 0x10,0x20,0x30,0x40,0x50

key1: 0x20,0xfe,0x9f,0x3c,0x57

key2: 0xab,0xcd,0xef,0x01,0x23

key3: 0x99,0x88,0x62,0x43,0x75

(4)wep3 コマンド: WEP を OFF にします。

>wep3

設定は recfg コマンド実行後に反映されます。

(5)frag コマンド: fragment Threshold を設定します。

>frag fragment_threshold

fragment_threshold には 256～2346 の偶数を指定してください。

設定は recfg コマンド実行後に反映されます。

(6)rts コマンド: RTS/CTS Threshold を設定します。

>rts rts/cts_threshold

rts/cts_threshold には 0～3000 の偶数を指定してください。
設定は recfg コマンド実行後に反映されます。

(7)gdt コマンド

>gdt rid_number

RID 番号を持つデータはこのコマンドで Dump 表示できます。

RID 番号は 16 進数で指定してください。

(8)sdt コマンド

>sdt rid_number

(入力例) Bold Italic が入力

>sdt FC00

Set WriteData

Record Length? ->2

DATA(WORD):0001

RID 番号を持つデータはこのコマンドでセットできます。

RID 番号は 16 進数で指定してください。

RecordLength は RID+DATA のワード数です。

データ数は指定した RecordLength-1 となります。

入力待ちで数値を何も入力せずに Enter を押すとその時点でこのコマンドは終了します。

指定した RID 番号が「StaticConfigurationEntities」に分類される場合、
recfg コマンド実行後に反映されます。

※本ドライバ内部変数に持っているわけではなく、Disable→Enable となった時点で
Farm 内に取り込まれるためです。

(8)recfg コマンド

>recfg

prism2 を一旦ディセーブルし、コンフィグレーションし、その後イネーブルします。
プロトコルスタックに関係なく設定変更を行います。

7. 制限

- (1)本ドライバは STATION 専用です。
- (2)PDA エリアの読み出し機能はサポートしていません。